Instalación HDFS

0.- Escenario

Instalaremos HDFS a mano

1.- Configuración de la red

Todos los equipos deben estar visibles entre ellos.

2.- Nombres de equipo

Hablar de Hadoop y HDFS es hablar de muchos equipos interconectados en clústers. Para simplificar la conexión, gestión y monitorización de todos los nodos es de mucha ayuda usar nombres significativos en lugar de las IPs.

```
sudo nano /etc/hostname
```

Con este comando editaremos el archivo hostname que es el que contiene el nombre del equipo local. Debemos modificar el nombre por alguno más descriptivo de la máquina como “master”, “slave01”, etc.

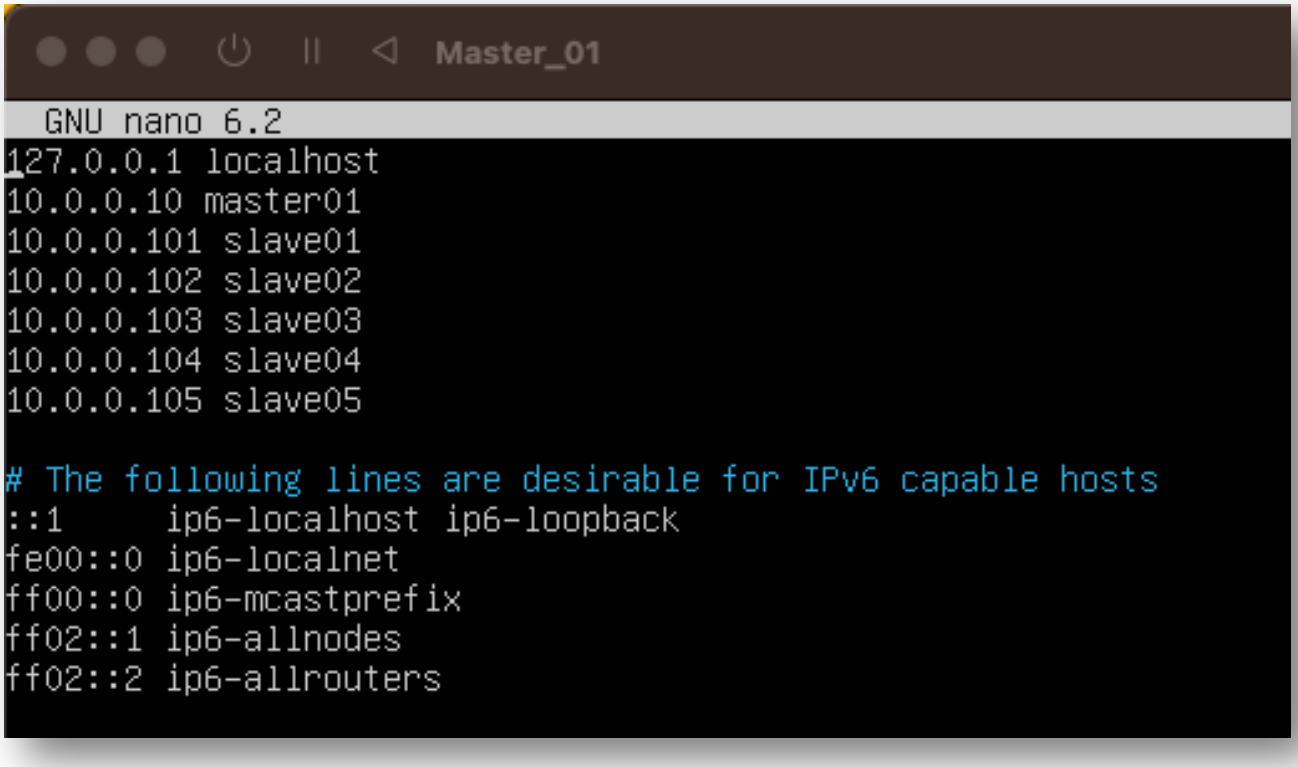
3.- DNS

Un servidor DNS resuelve nombre en direcciones IPs y viceversa. Para simplificar el escenario usaremos el archivo /etc/hosts que tenemos en el sistema operativo de cada máquina que actúa como una caché. Cuando el equipo quiere resolver un nombre se consultará este archivo en primer lugar. Si no pudiera resolver con la información que se guarda en este archivo entonces volverá a hacer una consulta pero esta vez a la IP configurada como nameserver.

```
sudo /etc/hosts
```

Con este comando editamos el archivo /etc/hosts para escribir una entrada de IP y nombre por cada línea.

En este escenario usaremos un equipo con el role de máster y otros cinco como esclavos por lo que mi archivo debería quedar de la siguiente manera:



La IP 127.0.0.1, que hace referencia al propio equipo, se mantiene y además añadimos las IPs junto con sus nombres de host que configuramos anteriormente. Este proceso es necesario repetirlo en cada máquina.

4.- Instalación JAVA

Muchas de las herramientas de Hadoop son aplicaciones JAVA por lo que necesitaremos el JRE para ejecutarlas o el JDK por si también necesitamos programar.

El problema que nos encontramos en esta parte es que muchas de las aplicaciones que usamos en Big Data ya son proyectos independientes de la fundación Apache. Un proyecto tiene autonomía de diseño y evolución por han ido evolucionando de maneras distintas y en muchos casos podemos tener incompatibilidades entre versiones mínimas y máximas necesarias de Java.

Ahora mismo la versión de JAVA más compatible es la 8 así que será esta la que instalemos.

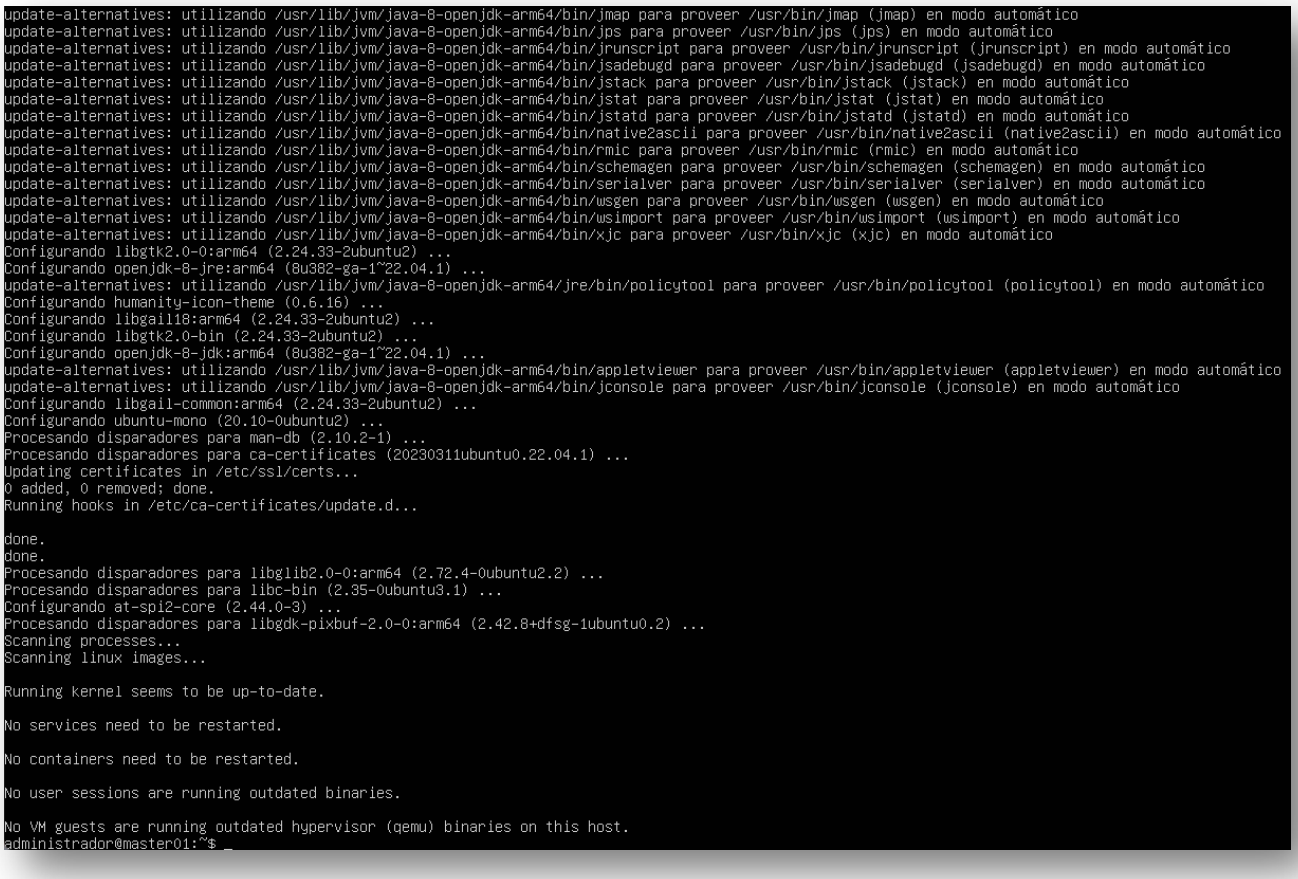
```
sudo apt-get install openjdk-8-jdk -y
```

Con este comando instalamos el paquete libre de JAVA en su versión 8, es una operación que tendremos que hacer en todos los equipos.

Es importante interpretar bien las salidas por pantalla de los comandos para saber si se ha ejecutado completamente y con éxito. No hay ninguna regla clara para saber si un comando se ha ejecutado bien, unas veces no hay ningún

mensaje como salida y otras veces aparecen muchas líneas. Siempre es necesario leer la salida y hacer una interpretación de la información que allí se muestre. En general cuando hay un error y el comando no se completa lo indica claramente.

Insisto, que la salida por pantalla después de ejecutar un comando tenga muchas líneas no quiere decir que el comando se haya ejecutado bien.

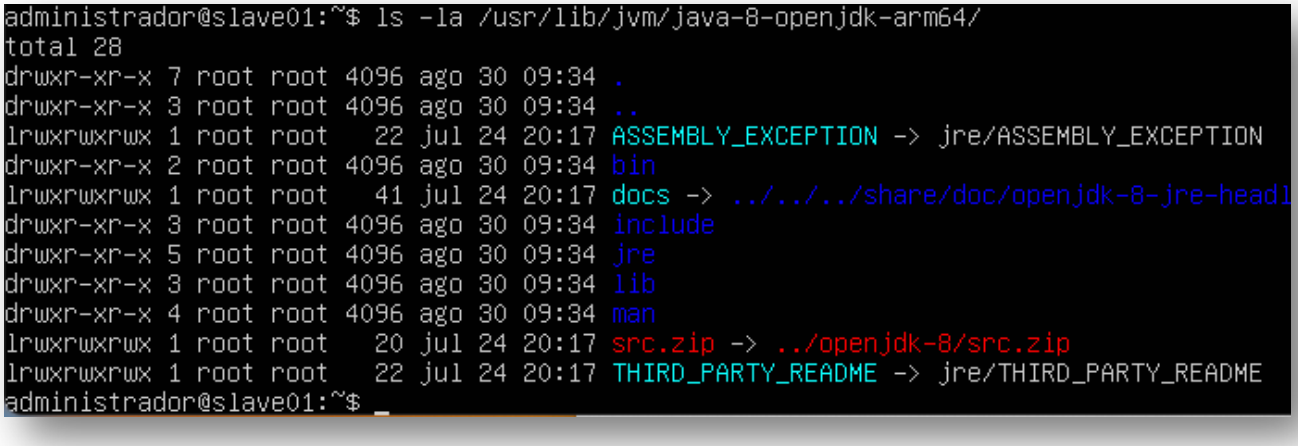


La captura anterior muestra una salida correcta. Si revisamos las líneas veremos que todas son informativas y que no muestran errores. Un par de líneas interesantes son las que empiezan por "**update-alternatives**" que indican cual es la versión de Java predeterminada. Esto es interesante en caso de tener varias versiones de Java en el sistema.

Otra de la información importante que podemos obtener de esa salida es la ruta donde se han guardado los archivos de esa versión de Java, en este caso en /usr/lib/jvm/java-8-openjdk-arm64

```
ls -la /usr/lib/jvm/java-8-openjdk-arm64
```

Con este comando listamos el contenido completo de la carpeta donde se ha instalado Java 8. Ojo con el copiar y pegar, el archivo que me estoy descargando es para la arquitectura ARM, probablemente tu equipo tenga arquitectura Intel y necesitarás descargar otro archivo.



En la captura anterior podemos ver la estructura de carpetas típica con los ejecutables, librerías, ayuda, etc de Java. Es habitual crear una variable de entorno en el sistema operativo llamada JAVA_HOME que apunte a esta carpeta de manera que cuando algún programa quiera lanzar la máquina virtual de Java solo tenga que llamar a esa variable.

```
sudo nano /etc/environment
```

Con este comando editamos el archivo de configuración de entorno global para incluir la variable y su valor. Debemos añadir al final del archivo la siguiente línea:

```
JAVA_HOME=”/usr/lib/jvm/java-8-openjdk-arm64”
```

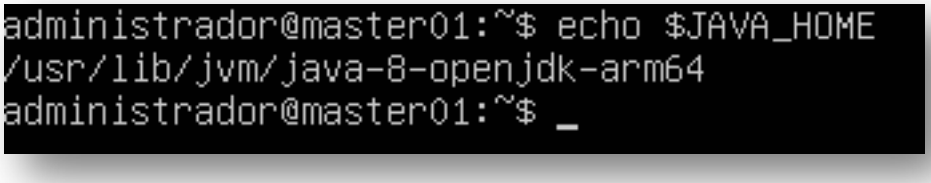
Guardamos el archivo y para que los cambios tengan efecto será necesario reiniciar el equipo. Si no fuera posible el reinicio por estar el equipo en otros usos podemos establecer esta variable temporalmente en esta sesión.

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-arm64
```

El comando anterior añade o modifica una variable de ambiente para la sesión actual. Este comando no genera ninguna salida si funciona correctamente. Aprovecho esta línea para hacer notar que es muy probable que la ruta de este ejemplo y vuestro caso particular difiera. En mi caso uso Ubuntu Server para plataformas ARM y seguramente en clase usemos Ubuntu Desktop para plataformas x64.

```
echo $JAVA_HOME
```

Con este comando imprimimos por pantalla el valor de una variable, fijaros en el uso del símbolo de dólar para hacer referencia al valor.



Si se muestra esta salida es que la variable existe y que tiene esa información.

5.- Servidor SSH

Algunas de las herramientas que usaremos requieren poder ejecutar comandos en equipos remotos por lo que necesitaremos el servicio SSH corriendo en cada máquina. La imagen de Ubuntu Server ya viene con un servidor SSH por defecto.

```
sudo apt-get install openssh-server
```

En caso de estar usando otra imagen, como por ejemplo un Ubuntu Desktop con interfaz gráfica, debemos instalar el servicio SSH con el comando anterior.

Las conexiones SSH requieren de autenticarse en el ordenador remoto mediante usuario y contraseña. Para hacer este proceso más transparente al usuario y aplicaciones podemos usar pares de claves públicas-privadas.

```
ssh-keygen -b 4096
```

Con el comando anterior generamos la pareja de llaves privada-pública. Con el modificador -b indicamos el número de bits de la clave.

Durante la ejecución del comando nos preguntarán por el nombre del archivo que guarda la clave, lo dejaremos por defecto. Hay que tener en cuenta que en entorno de producción tendremos llaves para muchos usuarios y que no las guardaremos en sus carpetas personales y sí en algún repositorio común.

También preguntará por si queremos añadir una contraseña con la que encriptar la llave privada de manera que nadie las pueda utilizar incluso si consiguen el archivo de llaves, podemos dejarlo vacío.

Este comando debemos repetirlo en cada una de las máquinas que usemos.

```
administrador@master01:~$ ssh-keygen -b 4096
Generating public/private rsa key pair.
Enter file in which to save the key (/home/administrador/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/administrador/.ssh/id_rsa
Your public key has been saved in /home/administrador/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:De24CikUOYOZVOejWdmWJut3NeMjZZ6um+IVaOO1AYQ administrador@master01
The key's randomart image is:
+---[RSA 4096]----+
|  ...=....=oo+.. |
|   .+..  *oE=    .|
|  .     + .=    ..|
|   .     =.     =+|
|  .     S.o   .=o+|
| .   .    .. ..*= |
|  . o    .   . =o..|
|   . . .    .o .. |
|      .    ...+o  |
+----[SHA256]-----+
administrador@master01:~$ _
```

La captura anterior muestra la salida del comando ssh-keygen si todo ha funcionado bien. Podemos ver la ruta y los nombres de la llave privada y la llave pública. Esta llave pública es la que tenemos que distribuir al resto de los equipos para que, en el momento en que intentemos una conexión a ese otro equipo, nuestra llave privada y esa llave pública que hemos distribuido.

```
administrador@master01:~$ ls -la .ssh
total 16
drwx------ 2 administrador administrador 4096 ago 30 15:12 .
drwxr-x--- 4 administrador administrador 4096 ago 25 21:29 ..
-rw------- 1 administrador administrador    0 ago 25 21:13 authorized_keys
-rw------- 1 administrador administrador 3389 ago 30 15:12 id_rsa
-rw-r--r-- 1 administrador administrador  748 ago 30 15:12 id_rsa.pub
administrador@master01:~$ _
```

Si comprobamos la ruta donde se guardan las llaves podemos ver los dos archivos. El que tiene la extensión pub es el que debemos distribuir en el resto de las máquinas. Para distribuirlo usaremos el siguiente comando:

```
ssh-copy-id -i /home/administrador/.ssh/id_rsa.pub administrador@slave01
```

El comando anterior será el encargado de conectarse al equipo remoto, slave01 en este ejemplo, usando su cuenta "administrador". Si es la primera vez que conectamos con ese equipo nos informará y pedirá confirmación para seguir conectando. Al conceder permiso nos preguntará por la contraseña de "administrador" en el equipo remoto. Al introducirla copiará nuestra llave pública en el equipo remoto. Desde ese momento podremos conectar a través de ssh con ese equipo remoto sin necesidad de introducir contraseña simplemente con el siguiente comando:

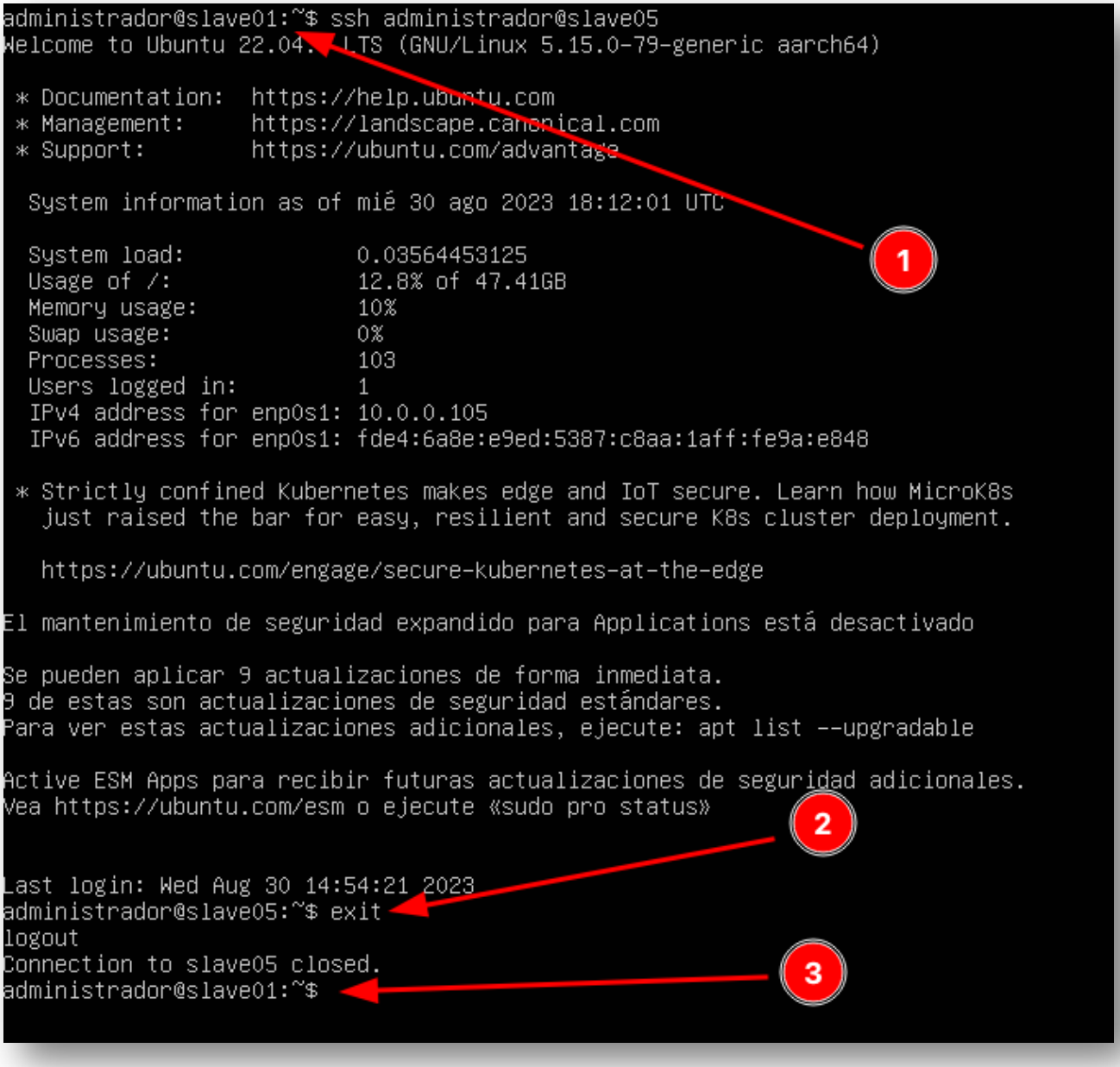
```
ssh administrador@slave01
```

Todas las llaves públicas se guardarán en el archivo .ssh/authorized_keys del equipo remoto.

Para completar la configuración de ssh tendremos que copiar la llave pública en todos los equipos de la red, incluso en el propio equipo ya que así también podremos usarlo como nodo de datos.

Llegados a este punto cualquier equipo puede conectarse por ssh a otro sin más que poner ssh usuario@equipo. Prueba a hacer alguna prueba como por ejemplo desde slave01 conectar a slave05 con el siguiente comando:

```
ssh administrador@slave05
```



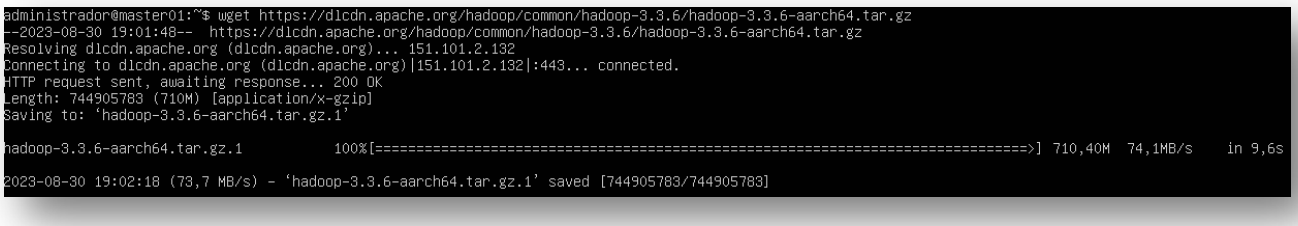
Si nos fijamos en esta captura, desde el prompt de slave01 nos conectamos a slave05. Nos muestra las líneas típicas de bienvenida y acaba en un prompt que nos dice que estamos en slave05. Por último cerramos la conexión con un "exit" y volvemos al prompt de slave01.

7.- Descarga Apache Hadoop

Desde el equipo que hará de máster descargamos la versión de Apache Hadoop desde la web oficial https://hadoop.apache.org/releases.html usando el siguiente comando:

```
wget https://dlcdn.apache.org/hadoop/common/hadoop-3.3.6/hadoop-3.3.6-aarch64.tar.gz
```

En este ejemplo particular estoy usando un Ubuntu Server ARM por lo que debo descargar el archivo de esa arquitectura, en este caso el que acaba en aarch64. El caso habitual es usar la arquitectura Intel cuyo archivo no tiene ningún sufijo especial.



La captura anterior indica que la descarga se ha realizado con éxito.

```
tar -xzf hadoop-3.3.6-aarch64.tar.gz
```

El comando anterior descomprime el archivo que indiquemos. Los modificadores indican que queremos extraer (x) usando el descompresor gzip (z) el contenido del archivo (f) que indicamos. Este comando no genera salida por pantalla pero si comprobamos nuevamente la carpeta veremos que hay un nuevo directorio llamado hadoop-3.3.6 con todos los archivos descomprimidos dentro.

```
mv hadoop-3.3.6 hadoop
```

No debería ser habitual tener varias versiones de Hadoop a la vez por lo que, para simplificar, vamos a cambiar el nombre de esa carpeta y dejarla solo hadoop.

Todos estos pasos anteriores será necesario repetirlos en el resto de los nodos o también podemos copiar la instalación que llevamos hecha hasta ahora.

```
rsync -rhv /home/administrador/hadoop/ administrador@slave01:/home/administrador/headoop
```

Repetiremos este comando por cada uno de los nodos que tengamos.

8.- Configuración Apache Hadoop CORE

En primer lugar, vamos a establecer unas variables de entorno para simplificar los archivos de configuración y así usar estas variables en lugar de escribir todas las rutas absolutas.

En un apartado anterior editamos directamente el archivo /etc/environment pero en esta ocasión vamos a editar alguno de los archivos de configuración y personalización que se encuentran ocultos en la carpeta personal del usuario. Aunque hay sutiles diferencias podemos elegir entre el archivo .bashrc y .profile que se encuentran en la carpeta del usuario. Ambos tienen como función personalizar el entorno para ese usuario.

A la hora de elegir una u otra opción para configurar las variables de entorno debes tener en cuenta que:

- /etc/environment afecta a todo el sistema y solo acepta declaración de variables sin recursividad.
- .bashrc y .profile solo afectan al usuario de su perfil y acepta programación tipo script.
- /etc/profiles afecta a todo el sistema y sí acepta recursividad en las variables

Discutiremos en clase ventajas e inconvenientes de ambas opciones, pero para esta guía vamos a probar en este punto editar el archivo .bashrc añadiendo al final de este las siguientes líneas.

```
nano /home/administrador/.bashrc
```

```
# Variables de entorno para Hadoop
export HADOOP_HOME=/home/administrador/hadoop
export PATH=$HADOOP_HOME/bin:$HADOOP_HOME/sbin:$PATH
```

Con estas líneas creamos una variable de entorno con la ruta absoluta de Hadoop y también añadimos al PATH las rutas de los ejecutables de Hadoop para que sean accesibles desde cualquier ruta del sistema.

Será necesario reiniciar el sistema o recargar el archivo .bashrc mediante el siguiente comando:

```
source /home/administrador.bashrc
```

La siguiente parte de la configuración se hará desde los archivos propios de Hadoop. Dentro de la carpeta de hadoop encontraremos otra llamada etc donde típicamente encontramos los archivos de configuración.

Todos estos pasos anteriores será necesario repetirlos en el resto de los nodos o también podemos copiar el archivo

```
rsync -hv /home/administrador/.bashrc administrador@slave01:/home/administrador/
```

Repetiremos este comando por cada uno de los nodos que tengamos.

El primer archivo de configuración que vamos a modificar es **/home/administrador/hadoop/etc/hadoop/core-site.xml** en el que indicaremos la dirección (ip+puerto) del equipo que actúa como NameNode. En nuestro caso será el equipo que llamamos master01. El archivo debe quedar como en la siguiente captura:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
  <property>
    <name>fs.defaultFS</name>
    <value>hdfs://master01:9000</value>
  </property>
</configuration>
```

Todos estos pasos anteriores será necesario repetirlos en el resto de los nodos o también podemos copiar el archivo

```
rsync -rhv /home/administrador/hadoop/etc/hadoop/core-site.xml administrador@slave01:/home/administrador/hadoop/etc/hadoop/
```

Repetiremos este comando por cada uno de los nodos que tengamos.

9.- Configuración Apache Hadoop HDFS

El siguiente archivo que vamos a editar es **/home/administrador/hadoop/etc/hadoop/hdfs-site.xml**. En este archivo indicaremos el nivel de replicación que queremos. Por defecto es de 3 copias pero nosotros lo vamos a cambiar a solo 2. En este archivo también vamos a indicar la ruta local donde se guardarán los datos HDFS, en este caso crearemos una carpeta en la ruta /home/administrador/hadoop/hdfs/namenode.

Por ahora es suficiente pero en este archivo también podremos modificar otros parámetros importantes como la lista de datanodes permitidos o no permitidos y el tamaño del bloque HDFS.

El archivo del namenode quedará como en la siguiente captura:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>

  <property>
    <name>dfs.replication</name>
    <value>2</value>
  </property>

  <property>
    <name>dfs.namenode.name.dir</name>
    <value>/home/administrador/hadoop/hdfs/namenode</value>
  </property>

</configuration>
```

Recuerda crear la ruta que has definido en el sistema de ficheros local de **namenode** con el siguiente comando:

```
mkdir /home/administrador/hadoop/hdfs/namenode -p
```

Usamos el parámetro -p para indicar que se deben crear los directorios padres hasta la ruta si estos no existieran.

En los datanode también modificaremos este archivo pero indicando la ruta donde guardará los archivos de HDFS.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>

  <property>
    <name>dfs.datanode.name.dir</name>
    <value>/home/administrador/hadoop/hdfs/datanode</value>
  </property>

</configuration>
```

Recuerda crear la ruta que has definido en el sistema de ficheros local de **datanode** con el siguiente comando:

```
mkdir /home/administrador/hadoop/hdfs/datanode -p
```

Usamos el parámetro -p para indicar que se deben crear los directorios padres hasta la ruta si estos no existieran.

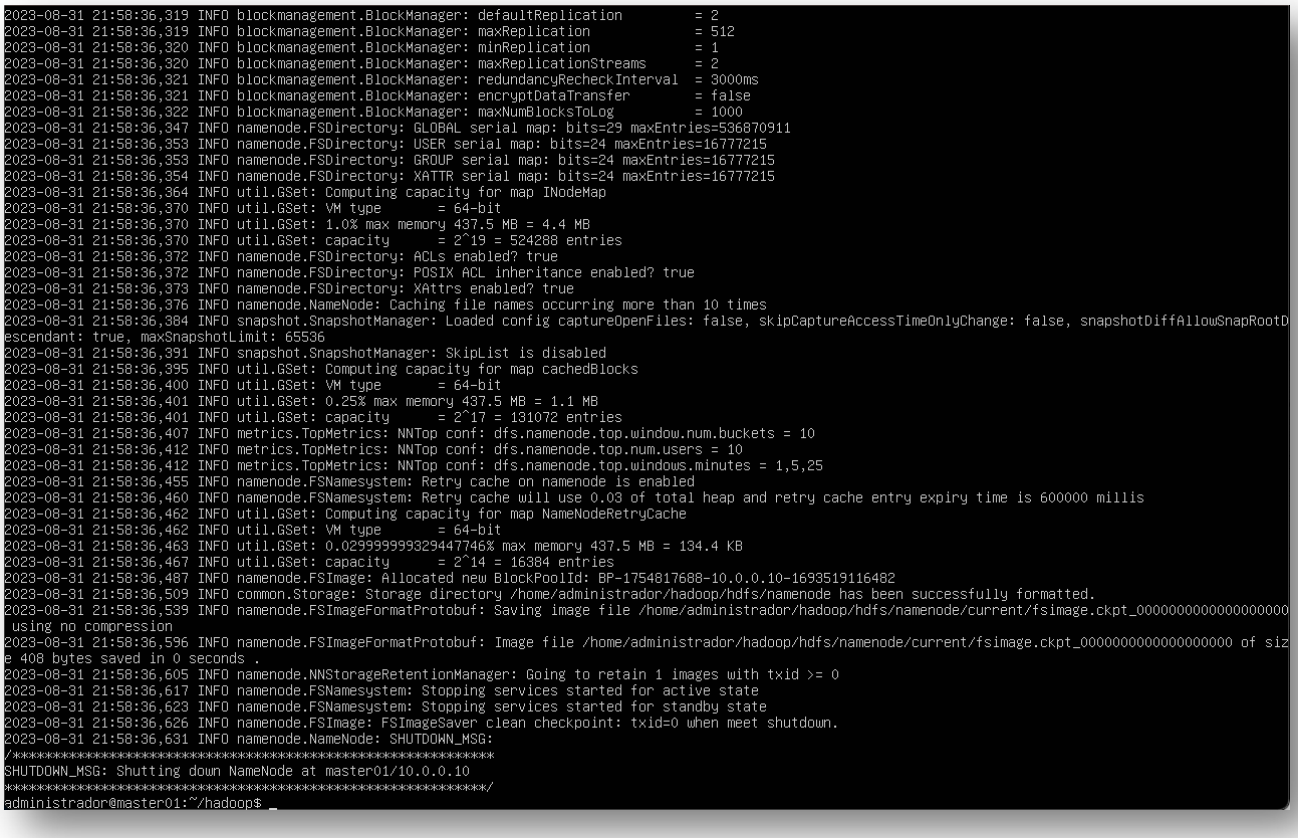
10.- Formatear HDFS

Desde el namenode ejecutamos el siguiente comando:

```
hdfs namenode -format
```

Este comando es el encargado de dar formato al sistema de ficheros HDFS. Provocará cambios en master01 que es el nodo que está funcionando como namenode.

La salida de este comando provoca muchas líneas de salida. Debería acaba como en la siguiente captura, sin mensajes de error.



11.- Arranque de HDFS

Una vez que ya tenemos formateado el sistema de ficheros HDFS es necesario activar los datanodes de manera que empiecen a funcionar y estar disponibles. Lo haremos con el siguiente comando:

```
start-dfs.sh
```

```
administrador@master01:~$ start-dfs.sh
Starting namenodes on [master01]
Starting datanodes
slave01: WARNING: /home/administrador/hadoop/logs does not exist. Creating.
slave05: WARNING: /home/administrador/hadoop/logs does not exist. Creating.
slave02: WARNING: /home/administrador/hadoop/logs does not exist. Creating.
slave03: WARNING: /home/administrador/hadoop/logs does not exist. Creating.
slave04: WARNING: /home/administrador/hadoop/logs does not exist. Creating.
Starting secondary namenodes [master01]
administrador@master01:~$ _
```

Podemos ver varios warnings que informan que la ruta por defecto donde se guardarán los logs de los datanode no existe pero que se crearán. También es interesante ver que ha creado un secondary namenode en el mismo nodo que el namenode.

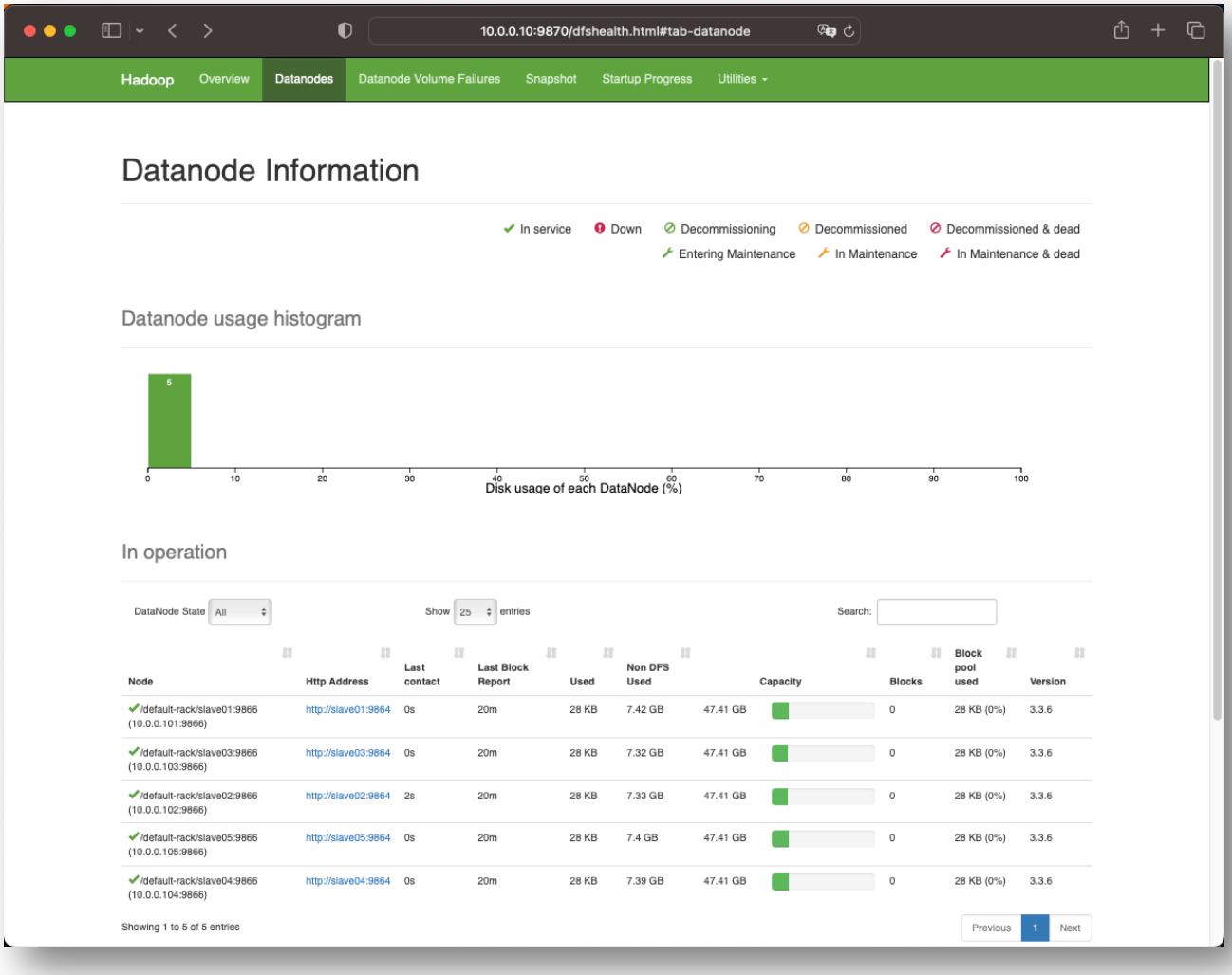
Para comprobar que todo está funcionando bien hay que comprobar qué máquinas virtuales Java están funcionando en cada nodo. Por ejemplo, en master01 tenemos funcionando el NameNode y el SecondaryNameNode.

En cambio si ejecutamos en slave01 solo veremos DataNode

```
administrador@master01:~$ jps
2690 NameNode
3044 Jps
2923 SecondaryNameNode
administrador@master01:~$
```

```
administrador@slave01:~$ jps
1996 DataNode
2079 Jps
administrador@slave01:~$
```

También es posible acceder a una interfaz gráfica vía web que ofrece HDFS en el namenode. Para ello abrimos un navegador en la URL http://10.0.0.10:9870 y desde allí consultar el estado de los nodos, navegar por el sistema de ficheros o consultar la configuración de HDFS.

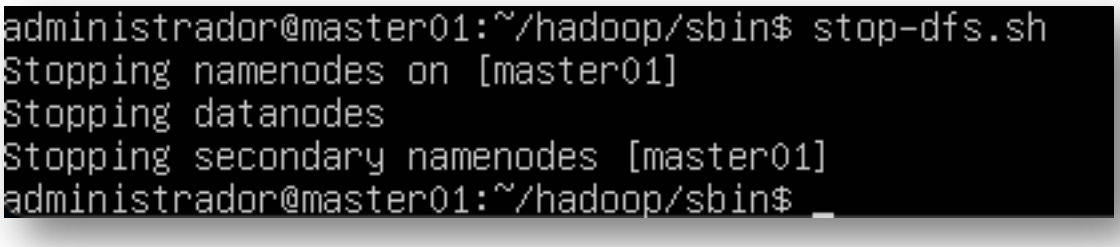


12.- Parada de HDFS

Para detener HDFS usaremos el siguiente comando desde el namenode, master01 en nuestro caso:

```
stop-dfs.sh
```

La salida del comando indicará que los datanodes, el namenode y el secondary namenode se han parado.



13.- Configuración Apache Hadoop YARN

El siguiente archivo que modificaremos es **/home/administrador/hadoop/etc/hadoop/yarn-site.xml** donde indicaremos el nombre del host del nodo maestro. También modificaremos los límites de uso de memoria en nuestras máquinas virtuales porque los valores por defecto esperan máquinas con mucha más capacidad que nuestras máquinas virtuales de 2 gigas de RAM.

En una configuración más completa será aquí donde determinemos las ACLs o la lista de nodemanager permitidos o no permitidos.

Por ahora lo dejaremos como en la siguiente captura:

```
<?xml version="1.0"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->
<configuration>

  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>master01</value>
  </property>

  <property>
    <name>yarn.nodemanager.resource.memory-mb</name>
    <value>1536</value>
  </property>

  <property>
    <name>yarn.scheduler.minimum-allocation-mb</name>
    <value>128</value>
  </property>

  <property>
    <name>yarn.scheduler-maximum-allocation-mb</name>
    <value>1536</value>
  </property>

  <property>
    <name>yarn.nodemanager.vmem-check-enabled</name>
    <value>false</value>
  </property>

</configuration>
```

También debemos modificar el archivo **/home/administrador/hadoop/etc/hadoop/workers** e incluir en él una lista de todos los nodos esclavos. Quedará como en la siguiente captura: